2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Distributed Function Calculation via Linear Iterations in the Presence of Malicious Agents – Part I: Attacking the Network

Shreyas Sundaram and Christoforos N. Hadjicostis

Abstract— **We consider the problem of distributed function calculation in the presence of faulty or malicious agents. In particular, we consider a setup where each node has an initial value and the goal is for (a subset of) the nodes to calculate a function of these values in a distributed manner. We focus on linear iterative strategies for function calculation, where each node updates its value at each time-step to be a weighted average of its own previous value and those of its neighbors; after a sufficiently large number of time-steps, each node is expected to have enough information to calculate the desired function of the initial node values. We study the susceptibility of such strategies to misbehavior by some nodes in the network; specifically, we consider a node to be malicious if it updates its value arbitrarily at each time-step, instead of following the predefined linear iterative strategy. If the connectivity of the network topology is $2f$ or less, we show that it is possible for a set of f malicious nodes to conspire in a way that makes it impossible for a subset of the other nodes in the network to correctly calculate an arbitrary function of all node values. Our analysis is constructive, in that it provides a specific scheme for the malicious nodes to follow in order to obfuscate the network in this fashion.**

I. Introduction

In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of certain parameters. For example, sink nodes in sensor networks may be tasked with calculating the average value of all the sensor measurements [1], [2]. Another example is the case of multi-agent systems, where all agents communicate with each other to coordinate their speed and direction [3]. The problem of function calculation in networks has been studied by the computer science, communication, and control communities over the past few decades, leading to the development of various protocols [4], [1], [5]. Special cases of distributed function calculation include data transmission from one or multiple sources to one or multiple sinks, and the *distributed consensus* problem, where all nodes in the network calculate the same function [4]. The notion of consensus has recently received extensive attention in the control literature, due to its applicability to cooperative control of multi-agent systems [6]. In these cases, the approach to consensus is to use a linear iteration, where each node in the network repeatedly updates its value to be a weighted linear combination of its own value and those of its neighbors (e.g., see [3], [6] and the references therein). These works have revealed that if the network topology satisfies certain conditions, the weights for the linear iteration can be chosen so that all of the nodes asymptotically converge to the same value. Recently, it was shown in [7], [8], [9] that this linear iterative strategy can actually be applied to the more general function calculation problem, allowing any node in the network to calculate any arbitrary function of the node values in a finite number of time-steps (upper bounded by the size of the network).

In this paper, we extend and generalize the above results on linear iterative strategies to address the problem of function calculation in the presence of malicious or faulty nodes. Specifically, we allow for the possibility that some nodes in the network update their values at each time-step in an arbitrary manner, instead of following the predefined strategy of using a specific weighted linear combination of their neighbors' (and own) values. Such arbitrary updates can occur, for example, if some nodes in the network are compromised by a malicious attacker whose objective is to disrupt the operation of the network [4], or they might be the result of hardware malfunctions at the nodes, which cause them to incorrectly calculate their update value [10]. The contribution of this paper is to show that the graph connectivity is a determining factor for the ability of linear iterative strategies to tolerate malicious (or faulty) agents. In particular, if the connectivity of the graph is $2f$ or less, then it is possible to find a subset of f nodes that can conspire to prevent some nodes from calculating an arbitrary function of all node values (regardless of the choice of weights in the linear iteration). This result has implications for the fault-tolerant distributed consensus problem, where all nodes are required to calculate the same function, even when there are a certain number of malicious nodes in the network. While we focus on the attacker's perspective in this paper, we also show in the companion paper [11] that linear iterative strategies can be made robust against malicious nodes if the connectivity of the network is sufficiently high. Together, these results effectively narrow the gap between linear iterative schemes and existing fault-tolerant consensus protocols (such as those described in [4]). It is worth noting that the recent paper [12] also considers the problem of reaching distributed consensus in the presence of malicious nodes through a model that is similar to the one considered in our work; however, that paper only requires the non-malicious nodes to asymptotically reach agreement on the same value (this value does not necessarily have to be

This material is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696 and NSF ITR Award 0426831. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

The authors are with the Coordinated Science Laboratory, and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL, 61801-2307, USA. E-mails: {ssundarm, chadjic}@uiuc.edu.

978-1-4244-2079-7/08/$25.00 ©2008 AACC.

any specific function of the initial values). In contrast, we consider the more general problem of enabling each node to calculate any arbitrary function of the initial values despite the presence of malicious nodes, and furthermore, for the special case of distributed consensus, our results allow the non-malicious nodes to reach consensus in finite-time.

In our development, we use $\mathbf{e}_i$ to denote the column vector with a 1 in its i–th position and 0's elsewhere. The symbol I_N denotes the $N \times N$ identity matrix, and the notation A' indicates the transpose of matrix A. We will denote the rank of matrix A by $\rho(A)$, and we will denote the column space of matrix A by $\mathcal{R}(A)$. We will also denote the cardinality of a set S by $|S|$, and for a pair of sets $\mathcal{S}$ and $\mathcal{T}$, we will use $\mathcal{S}-\mathcal{T}$ to denote the set of elements of $\mathcal{S}$ that are not in $\mathcal{T}$.

II. Background on Graph Theory

We will use the following terminology in our discussion. Further details can be found in standard texts on graph theory, such as [13].

A graph is an ordered pair $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is a set of vertices, and $\mathcal{E}$ is a set of ordered pairs of vertices, called directed edges. If $(x_i, x_j) \in \mathcal{E} \Leftrightarrow (x_j, x_i) \in \mathcal{E}$, the graph is said to be undirected. The nodes in the set $\mathcal{N}_i = \{x_j | (x_j, x_i) \in \mathcal{E}\}$ are said to be neighbors of node i, and the in-degree of node i is denoted by $\deg_i = |\mathcal{N}_i|$. A *subgraph* of $\mathcal{G}$ is a graph $\mathcal{H} = \{\bar{\mathcal{X}}, \bar{\mathcal{E}}\}$, with $\bar{\mathcal{X}} \subseteq \mathcal{X}$ and $\bar{\mathcal{E}} \subseteq \mathcal{E}$ (where all edges in $\bar{\mathcal{E}}$ are between vertices in $\bar{\mathcal{X}}$). A subgraph $\mathcal{H}$ of $\mathcal{G}$ is said to be *induced* if, whenever $x_i, x_j \in \bar{\mathcal{X}}$, $(x_i, x_j) \in \bar{\mathcal{E}} \Leftrightarrow (x_i, x_j) \in \mathcal{E}$.

A *path* P from vertex x_{i_0} to vertex x_{i_t} is a sequence of vertices $x_{i_0}, x_{i_1}, \ldots, x_{i_t}$ such that $(x_{i_j}, x_{i_{j+1}}) \in \mathcal{E}$ for $0 \leq j \leq t-1$. A path is called a *cycle* if its start vertex and end vertex are the same, and no other vertex appears more than once in the path. Paths P_1 and P_2 are *vertex disjoint* if they have no vertices in common. A set of paths $P_1, P_2, \ldots, P_r$ are vertex disjoint if the paths are pairwise vertex disjoint. Given two subsets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$, a set of r vertex disjoint paths, each with start vertex in $\mathcal{X}_1$ and end vertex in $\mathcal{X}_2$, is called an *r-linking* from $\mathcal{X}_1$ to $\mathcal{X}_2$. Note that if $\mathcal{X}_1$ and $\mathcal{X}_2$ are not disjoint, we will take each of their common vertices to be a vertex disjoint path between $\mathcal{X}_1$ and $\mathcal{X}_2$ of length zero.

A graph is said to be *strongly connected* if there is a path between vertices x_i to x_j for every $x_i, x_j \in \mathcal{X}$. We will call a graph *disconnected* if there exists at least one pair of vertices $x_i, x_j \in \mathcal{X}$ such that there is no path from x_i to x_j. A *vertex cut* in a graph is a subset $\mathcal{S} \subset \mathcal{X}$ such that removing the vertices in $\mathcal{S}$ (and the associated edges) from the graph causes the graph to be disconnected. A graph is said to be κ-connected if every vertex cut has cardinality at least κ. The *connectivity* of a graph is the smallest size of a vertex cut. Note that if a graph is κ-connected, the in-degree of every node must be at least κ (otherwise, we can disconnect the graph by removing all the neighbors of the offending node, thereby producing a vertex cut of size less than κ).

III. Function Calculation via Linear Iterations

The interaction constraints in distributed systems and networks can be conveniently modeled via a directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ represents the communication constraints in the network (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information directly from node x_j). Note that undirected graphs can be readily handled by treating each undirected edge as two directed edges.

Suppose that each node i has some initial value, given by $x_i[0]$, and the goal is for (a subset of) the nodes to calculate some function of these initial values. At each time-step k, all nodes can update and/or exchange their values based on some strategy that adheres to the constraints imposed by the network topology. The scheme that we study in this paper makes use of linear iterations; specifically, at each time-step, each node updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{j \in \mathcal{N}_i} w_{ij} x_j[k] \ , \qquad (1)$$

where the w_{ij}'s are a set of weights.[1] In other words, each node updates its value to be a linear combination of its own value and the values of its neighbors. For ease of analysis, the values of all nodes at time-step k can be aggregated into the value vector $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}'$, and the update strategy for the entire system can be represented as $x[k+1] = Wx[k]$, for $k = 0, 1, \ldots$, where the (i,j)–th entry of the weight matrix W is the weight w_{ij} (note that $w_{ij} = 0$ if $j \notin \mathcal{N}_i$). The values (or outputs) that are available to node i during the k–th time-step will be denoted by $\mathbf{y}_i[k] = C_i \mathbf{x}[k]$, where C_i is a $(\deg_i + 1) \times N$ matrix with a single 1 in each row denoting the positions of the state-vector $\mathbf{x}[k]$ that are available to node i (i.e., these positions correspond to nodes that are neighbors of node i, along with node i itself).

Definition 1: Let $g : \mathbb{R}^N \mapsto \mathbb{R}^q$ be a function of the initial values of the nodes (note that $g(\cdot)$ will be a vector-valued function if $q \geq 2$). We say $g(x_1[0], x_2[0], \ldots, x_N[0])$ is *calculable by node i* if it can be calculated by node i after running the linear iteration for a sufficiently large number of time-steps. We call $g(x_1[0], x_2[0], \ldots, x_N[0])$ a *linear function* if it is of the form $Qx[0]$ for some $q \times N$ matrix Q. The system is said to achieve *distributed consensus* if all nodes in the system calculate the same function $g(x_1[0], x_2[0], \ldots, x_N[0])$ after running the linear iteration for a sufficiently large number of time-steps. □

In [9], it was shown that, for almost any choice of weights, the nodes in the system can calculate any arbitrary function of the other node values after running the linear iteration $x[k+1] = Wx[k]$ for a finite number of time-steps (as long as there are paths from the nodes that hold the needed values to the nodes that have to calculate the functions). We will now summarize the salient points of the analysis in that paper. First, by noting that $\mathbf{x}[k] = W^k \mathbf{x}[0]$, the output at time-step k can be written as $\mathbf{y}_i[k] = C_i W^k \mathbf{x}[0]$, and the

[1] The methodology for choosing the weights appropriately and the implications of this choice are discussed later in the paper.

set of all outputs seen by node i over $L+1$ time-steps is given by

$$\underbrace{\begin{bmatrix}\mathbf{y}_i[0]\\ \mathbf{y}_i[1]\\ \vdots \\ \mathbf{y}_i[L]\end{bmatrix}}_{\mathbf{y}_i[0:L]} = \underbrace{\begin{bmatrix}C_i\\ C_iW\\ \vdots \\ C_iW^L\end{bmatrix}}_{\mathcal{O}_{i,L}} \mathbf{x}[0] \ . \tag{2}$$

When $L = N-1$, the matrix $\mathcal{O}_{i,L}$ in the above equation is the *observability matrix* for the pair (W, C_i) [14]. The row-space of $\mathcal{O}_{i,L}$ characterizes the set of all linear functions of $\mathbf{x}[0]$ that can be calculated by node i up to time-step L. Specifically, if the row space of the observability matrix $\mathcal{O}_{i,L}$ contains a matrix Q, one can find a matrix Γ_i such that $\Gamma_i\mathcal{O}_{i,L} = Q$. Thus, after running the linear iteration for $L+1$ time-steps, node i can immediately calculate the linear function $Q\mathbf{x}[0]$ as a linear combination of the outputs of the system over those time steps, i.e.,

$$\Gamma_i\mathbf{y}_i[0:L] = \Gamma_i\mathcal{O}_{i,L}\mathbf{x}[0] = Q\mathbf{x}[0] \ . \tag{3}$$

If $\rho(\mathcal{O}_{i,L}) = N$, the pair (W, E_i) is said to be *observable*. In this case, node i can determine the entire initial value vector $\mathbf{x}[0]$ from the outputs of the system (since the matrix $Q = I_N$ will be contained in the row space of $\mathcal{O}_{i,L}$), and can therefore calculate any function of those values.

An important feature of the observability matrix is that there exists an integer ν_i such that $\rho(\mathcal{O}_{i,0}) < \rho(\mathcal{O}_{i,1}) < \cdots < \rho(\mathcal{O}_{i,\nu_i-1}) = \rho(\mathcal{O}_{i,\nu_i}) = \rho(\mathcal{O}_{i,\nu_i+1}) = \cdots$. In other words, the rank of the matrix $\mathcal{O}_{i,L}$ monotonically increases with L until $L = \nu_i - 1$, at which point it stops increasing. This means that the outputs of the system $\mathbf{y}_i[0], \mathbf{y}_i[1], \ldots, \mathbf{y}_i[\nu_i - 1]$ contain the maximum amount of information that is possible to obtain about the initial state, and future outputs of the system do not provide any extra information to node i. The integer ν_i is called the *observability index* of the pair (W, C_i), and can be upper bounded as $\nu_i \leq N - \deg_i$ [9]. This implies that if it is possible for node i to calculate the desired value $g(x_1[0], \ldots, x_N[0])$, it can do so in at most $N - \deg_i$ time-steps.

The following theorem from [9] indicates that, for almost any choice of weight matrix, the observability matrix for each node i will allow node i to obtain the initial value of all nodes that have a path in the network to node i. As a consequence, each node i can calculate any arbitrary function of these initial values after running the linear iteration for a finite number of time-steps.

Theorem 1: Let $\mathcal{G}$ denote the graph of the network. Define the set $R_i = \{x_j |$ There exists a path from x_j to x_i in $\mathcal{G}\}$. Then, for almost any choice of weight matrix W, node i can obtain the value $x_j[0]$, $x_j \in R_i$, after running the linear iteration $x[k+1] = Wx[k]$ for $L_i + 1$ time-steps, for some $0 \leq L_i < |R_i| - \deg_i$; node i can therefore calculate any arbitrary function of the values $\{x_j[0] \mid x_j \in R_i\}$. □

In the above theorem, the phrase "almost any" indicates that the set of parameters for which the theorem does not hold has Lebesgue measure zero [9]. When the graph is strongly connected, there is a path from every node to every other node, and so each node can calculate any arbitrary function of the initial values after running the linear iteration for $\max_i(N - \deg_i)$ time-steps. As discussed in [9], the weights can be chosen (almost arbitrarily) by a centralized entity and provided to the nodes[2] *a priori*, or they can be chosen independently by each node and discovered by the network after following a simple distributed protocol.

Remark 1: Note that unlike asymptotic consensus schemes, where $\mathbf{x}[k]$ converges to a constant vector after running the linear iteration for an infinite number of time-steps, the protocol described above *does not* require $\mathbf{x}[k]$ to converge to any particular vector (or even to converge at all). Instead, each node i is able to calculate its desired function from (3) by examining the *evolution* of its own values and the values of its neighbors over a finite number of time-steps. □

In this paper, we will examine the susceptibility of linear iteration based function calculation schemes to misbehavior by a set of nodes that update their values at each time-step in a malicious manner.

IV. Modeling Malicious Nodes and Main Result

Suppose the objective in the system is for each node i to calculate $g_i(x_1[0], x_2[0], \ldots, x_N[0])$, for some function $g_i : \mathbb{R}^N \rightarrow \mathbb{R}^{q_i}$ that could be different for each node. When there are no malicious nodes in the network, we saw in the last section that this can be accomplished by having the nodes run the linear iteration $x[k+1] = Wx[k]$ with almost any weight matrix W for a finite number of time-steps. Suppose, however, that instead of applying the update equation (1), some node l updates its value at each time-step as

$$x_l[k+1] = w_{ll}x_l[k] + \sum_{j \in \mathcal{N}_l} w_{lj}x_j[k] + u_l[k] \ , \tag{4}$$

where $u_l[k]$ is an additive error at time-step k.

Definition 2: Suppose all nodes run the linear iteration for T time-steps in order to perform function calculation. Node l is said to be *malicious* (or *faulty*) if $u_l[k]$ is nonzero for at least one time-step k, $0 \leq k \leq T-1$. □

Note that the model for malicious nodes considered here is quite general, and allows node l to update its value in a completely arbitrary manner (via appropriate choices of the error $u_l[k]$ at each time-step). Let $\mathcal{S} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_f}\}$ denote the set of nodes that are malicious during a run of the linear iteration. Using (4), the linear iteration can then be modeled as

$$\mathbf{x}[k+1] = W\mathbf{x}[k] + \underbrace{\begin{bmatrix}\mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \cdots & \mathbf{e}_{i_f}\end{bmatrix}}_{B_{\mathcal{S}}} \underbrace{\begin{bmatrix}u_{i_1}[k]\\ u_{i_2}[k]\\ \vdots \\ u_{i_f}[k]\end{bmatrix}}_{\mathbf{u}_{\mathcal{S}}[k]}$$

$$\mathbf{y}_i[k] = C_i\mathbf{x}[k], \ \ 1 \leq i \leq N \ , \tag{5}$$

[2] Actually, each node i only requires the weights corresponding to the i-th row of W, along with the coefficient matrix Γ_i solving (3) with $Q = I_N$.

where $\mathbf{y}_i[k]$ represents the outputs (node values) seen by node i during time-step k of the linear iteration (recall that C_i is a $(\deg_i+1)\times N$ matrix with a single 1 in each row capturing the positions of the state-vector $\mathbf{x}[k]$ that are available to node i, and $\mathbf{e}_l$ denotes a unit vector with a single nonzero entry with value 1 at its l–th position). The set of all values seen by node i during the first $L+1$ time-steps of the linear iteration is given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ C_iB_{\mathcal{S}} & 0 & \cdots & 0 \\ C_iWB_{\mathcal{S}} & C_iB_{\mathcal{S}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_iW^{L-1}B_{\mathcal{S}} & C_iW^{L-2}B_{\mathcal{S}} & \cdots & C_iB_{\mathcal{S}} \end{bmatrix}}_{\mathcal{M}_{i,L}^{\mathcal{S}}} \underbrace{\begin{bmatrix} \mathbf{u}_{\mathcal{S}}[0] \\ \mathbf{u}_{\mathcal{S}}[1] \\ \mathbf{u}_{\mathcal{S}}[2] \\ \vdots \\ \mathbf{u}_{\mathcal{S}}[L-1] \end{bmatrix}}_{\mathbf{u}_{\mathcal{S}}[0:L-1]}, \tag{6}$$

where $\mathbf{y}_i[0:L]$ and $\mathcal{O}_{i,L}$ are defined in equation (2). The matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{S}}$ will characterize the ability of node i to calculate the required function of the initial values, and we will call $\mathcal{M}_{i,L}^{\mathcal{S}}$ the *fault matrix* for the triplet $(W, B_{\mathcal{S}}, C_i)$. In our development, we will use the fact that matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{S}}$ can be expressed recursively as

$$\mathcal{O}_{i,L} = \begin{bmatrix} C_i \\ \mathcal{O}_{i,L-1}W \end{bmatrix}, \quad \mathcal{M}_{i,L}^{\mathcal{S}} = \begin{bmatrix} 0 & 0 \\ \mathcal{O}_{i,L-1}B_{\mathcal{S}} & \mathcal{M}_{i,L-1}^{\mathcal{S}} \end{bmatrix}, \tag{7}$$

where $\mathcal{O}_{i,0} = C_i$ and $\mathcal{M}_{i,0}^{\mathcal{S}}$ is the empty matrix (with zero columns). We will demonstrate the following key result, showing how a set of malicious nodes can prevent some nodes in the network from calculating an arbitrary function of all initial node values.

Theorem 2: Let the graph of the given network $\mathcal{G}$ have connectivity κ. If $\kappa \leq 2f$, then regardless of the choice of weight matrix in the linear iterative strategy, it is possible for f malicious nodes to conspire to update their values in such a way that some node cannot correctly calculate an arbitrary function of all initial node values, regardless of the number of time-steps for which the linear iteration is run. □

We will develop the proof of this theorem over the remainder of the paper. Note that naturally, there is no way to prevent a malicious node from trying to influence the result of a computation by changing its own *initial* value. We will choose not to address this here, because of the philosophically different nature of this issue, and because of the fact that our problem formulation remains valid in cases where malicious nodes do not contribute initial values (i.e., they function as routers).

V. Attacking the Network When $\kappa \leq 2f$

In order to prove Theorem 2, we will start by establishing a relationship between the column space of the fault matrices and the column space of the observability matrix for certain nodes in the network. To do this, consider the graph of a given network $\mathcal{G}$, and let $\mathcal{S}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|\mathcal{S}_1|}}\}$, $\mathcal{S}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|\mathcal{S}_2|}}\}$ denote disjoint sets of vertices such that $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ forms a vertex cut of $\mathcal{G}$. Let $x_i, x_j \in \mathcal{X} - \mathcal{S}$ be nodes such that there is no path from node j to node i in the graph induced by $\mathcal{X} - \mathcal{S}$ (such nodes exist because $\mathcal{S}$ is a vertex cut). Let $\mathcal{H}$ denote the set of all nodes that have a path to node i in the graph induced by $\mathcal{X} - \mathcal{S}$, and let $\bar{\mathcal{H}} = \mathcal{X} - (\mathcal{H} \cup \mathcal{S})$.

Theorem 3: For any nonnegative integer L, the columns of the observability matrix $\mathcal{O}_{i,L}$ corresponding to the nodes in $\bar{\mathcal{H}}$ can be written as a linear combination of the columns in the matrices $\mathcal{M}_{i,L}^{\mathcal{S}_1}$ and $\mathcal{M}_{i,L}^{\mathcal{S}_2}$. □

Proof: Let $\mathbf{x}_{\mathcal{H}}[k]$ denote the vector of values of nodes in set $\mathcal{H}$, $\mathbf{x}_{\mathcal{S}_1}[k]$ denote the vector of values of nodes in set $\mathcal{S}_1$, $\mathbf{x}_{\mathcal{S}_2}[k]$ denote the vector of values of nodes in set $\mathcal{S}_2$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vector of values of nodes in set $\bar{\mathcal{H}}$. Note that $x_i[k]$ is contained in $\mathbf{x}_{\mathcal{H}}[k]$, $x_j[k]$ is contained in $\mathbf{x}_{\bar{\mathcal{H}}}[k]$, and that the sets $\mathcal{H}, \mathcal{S}_1, \mathcal{S}_2$, and $\bar{\mathcal{H}}$ are disjoint. Assume without loss of generality that the vector $\mathbf{x}[k]$ in (5) is of the form $\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{S}_1}[k] & \mathbf{x}'_{\mathcal{S}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k] \end{bmatrix}'$ (it can always be put into this form via an appropriate permutation of the node indices). Then, since no node in set $\mathcal{H}$ has an incoming edge from any node in set $\bar{\mathcal{H}}$ (otherwise, there would be a path from a node in $\bar{\mathcal{H}}$ to node i), the weight matrix for the linear iteration must necessarily have the form

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} & 0 \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix}. \tag{8}$$

The C_i matrix in (5) for node i must be of the form $C_i = \begin{bmatrix} C_{i,1} & C_{i,2} & C_{i,3} & 0 \end{bmatrix}$, again because node i has no neighbors in set $\bar{\mathcal{H}}$. Furthermore, from the definition of the matrix $B_{\mathcal{S}}$ in (5), note that this ordering of nodes implies that $B_{\mathcal{S}_1} = \begin{bmatrix} 0 & I_{|\mathcal{S}_1|} & 0 & 0 \end{bmatrix}'$, $B_{\mathcal{S}_2} = \begin{bmatrix} 0 & 0 & I_{|\mathcal{S}_2|} & 0 \end{bmatrix}'$. Let n denote the number of nodes in set $\bar{\mathcal{H}}$ (i.e., $\mathbf{x}_{\bar{\mathcal{H}}}[k] \in \mathbb{R}^n$). For any nonnegative integer L, the set of columns of the observability matrix $\mathcal{O}_{i,L}$ corresponding to the nodes in $\bar{\mathcal{H}}$ is given by $\mathcal{O}_{i,L}\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix}$. Using the recursive definition of $\mathcal{O}_{i,L}$ in (7), and the fact that $C_i\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix} = 0$, we obtain

$$\begin{aligned} \mathcal{O}_{i,L}\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix} &= \begin{bmatrix} C_i \\ \mathcal{O}_{i,L-1}W \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{O}_{i,L-1} \end{bmatrix} W \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \mathcal{O}_{i,L-1} \end{bmatrix} B_{\mathcal{S}_1}W_{24} + \begin{bmatrix} 0 \\ \mathcal{O}_{i,L-1} \end{bmatrix} B_{\mathcal{S}_2}W_{34} \\ &\quad + \begin{bmatrix} 0 \\ \mathcal{O}_{i,L-1} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix} W_{44}. \end{aligned}$$

Applying the above procedure recursively for matrices of the form $\begin{bmatrix} 0 \\ \mathcal{O}_{i,L-\alpha} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ I_n \end{bmatrix}$, $1 \leq \alpha \leq L$, we obtain (after some

algebraic manipulation)

$$\mathcal{O}_{i,L}\begin{bmatrix}0\\0\\0\\I_n\end{bmatrix} = \mathcal{M}_{i,L}^{\mathcal{S}_1}\begin{bmatrix}W_{24}\\W_{24}W_{44}\\\vdots\\W_{24}W_{44}^{L-1}\end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{S}_2}\begin{bmatrix}W_{34}\\W_{34}W_{44}\\\vdots\\W_{34}W_{44}^{L-1}\end{bmatrix}. \tag{9}$$

This concludes the proof of the theorem. ■

We now show how a certain set of nodes can maliciously update their values so that some node i cannot obtain any information about the initial values of some other nodes in the network.

Lemma 1: If nodes in set $\mathcal{S}_1$ are malicious, it is possible for them to update their values in such a way that the values seen by node i (over any number of time-steps of the linear iteration) are indistinguishable from the values seen by node i when nodes in set $\mathcal{S}_2$ are malicious. Furthermore, these indistinguishable faults make it impossible for node i to determine the initial values of nodes in the set $\bar{\mathcal{H}}$.

Proof: As in the proof of Theorem 3, let $\mathbf{x}_{\mathcal{H}}[k], \mathbf{x}_{\mathcal{S}_1}[k]$, $\mathbf{x}_{\mathcal{S}_2}[k]$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vector of values of nodes in sets $\mathcal{H}$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\bar{\mathcal{H}}$, respectively, and assume (without loss of generality) that the vector $\mathbf{x}[k]$ in (5) is of the form $\mathbf{x}[k] = \begin{bmatrix}\mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{S}_1}[k] & \mathbf{x}'_{\mathcal{S}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k]\end{bmatrix}'$. Let n be the number of nodes in set $\bar{\mathcal{H}}$ and let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ be arbitrary vectors. We will now show that the values seen by node i when nodes in $\mathcal{S}_1$ are malicious and $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$ will be indistinguishable from the values seen by node i when nodes in $\mathcal{S}_2$ are malicious and $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$. This will imply that node i cannot determine whether the initial values of nodes in $\bar{\mathcal{H}}$ are given by vector $\mathbf{a}$ or vector $\mathbf{b}$.

To this end, suppose the nodes in set $\mathcal{S}_1$ are malicious. From (6), the values seen by node i over $L+1$ time-steps are given by $\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{S}_1}\mathbf{u}_{\mathcal{S}_1}[0:L-1]$. From Theorem 3 (specifically, equation (9)), this expression can be written as

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\begin{bmatrix}\mathbf{x}_{\mathcal{H}}[0]\\\mathbf{x}_{\mathcal{S}_1}[0]\\\mathbf{x}_{\mathcal{S}_2}[0]\\0\end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{S}_1}\mathbf{u}_{\mathcal{S}_1}[0:L-1] + \left(\mathcal{M}_{i,L}^{\mathcal{S}_1}\begin{bmatrix}W_{24}\\W_{24}W_{44}\\\vdots\\W_{24}W_{44}^{L-1}\end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{S}_2}\begin{bmatrix}W_{34}\\W_{34}W_{44}\\\vdots\\W_{34}W_{44}^{L-1}\end{bmatrix}\right)\mathbf{x}_{\bar{\mathcal{H}}}[0]. \tag{10}$$

Suppose $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$, and that nodes in $\mathcal{S}_1$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{S}_1}[k] = W_{24}W_{44}^k(\mathbf{b}-\mathbf{a})$, producing the error vector

$$\mathbf{u}_{\mathcal{S}_1}[0:L-1] = \begin{bmatrix}W_{24}\\W_{24}W_{44}\\\vdots\\W_{24}W_{44}^{L-1}\end{bmatrix}(\mathbf{b}-\mathbf{a}) \ . \tag{11}$$

Substituting this into the expression for $\mathbf{y}_i[0:L]$ (with $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$), the values seen by node i under this fault scenario are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\begin{bmatrix}\mathbf{x}_{\mathcal{H}}[0]\\\mathbf{x}_{\mathcal{S}_1}[0]\\\mathbf{x}_{\mathcal{S}_2}[0]\\0\end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{S}_1}\begin{bmatrix}W_{24}\\W_{24}W_{44}\\\vdots\\W_{24}W_{44}^{L-1}\end{bmatrix}\mathbf{b} + \mathcal{M}_{i,L}^{\mathcal{S}_2}\begin{bmatrix}W_{34}\\W_{34}W_{44}\\\vdots\\W_{34}W_{44}^{L-1}\end{bmatrix}\mathbf{a} \ . \tag{12}$$

Now suppose that nodes in $\mathcal{S}_2$ are malicious (instead of nodes in $\mathcal{S}_1$). Again, from (6) and Theorem 3, the values seen by node i over $L+1$ time-steps will be given by equation (10), except with the term $\mathcal{M}_{i,L}^{\mathcal{S}_1}\mathbf{u}_{\mathcal{S}_1}[0:L-1]$ replaced by $\mathcal{M}_{i,L}^{\mathcal{S}_2}\mathbf{u}_{\mathcal{S}_2}[0:L-1]$. If $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$, and nodes in $\mathcal{S}_2$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{S}_2}[k] = W_{34}W_{44}^k(\mathbf{a}-\mathbf{b})$, the set of values seen by node i will be identical to the expression in (12), and thus the values received by node i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$ and the nodes in $\mathcal{S}_1$ are malicious will be indistinguishable from the values seen by node i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$ and the nodes in $\mathcal{S}_2$ are malicious. Since this holds for all nonnegative integers L, this fault scenario makes it impossible for node i (and in fact, any node in set $\mathcal{H}$) to obtain the initial values of node j (or any other node in set $\bar{\mathcal{H}}$). ■

We are now in place to prove the main theorem of the paper (Theorem 2, given at the end of Section IV).

Proof: [Theorem 2] In Lemma 1, we saw that if the union of the disjoint sets of vertices $\mathcal{S}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|S_1|}}\}$, $\mathcal{S}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|S_2|}}\}$ forms a vertex cut, then some node i cannot distinguish a particular set of errors by nodes in $\mathcal{S}_1$ from another set of errors by nodes in $\mathcal{S}_2$. Furthermore, these errors make it impossible for node i to obtain any information about the initial values of some other nodes in the network (i.e., node i cannot determine whether the initial values of some other nodes are given by $\mathbf{a}$ or $\mathbf{b}$, for some vectors $\mathbf{a}$ and $\mathbf{b}$). Choose $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $|S_1| = \lfloor \frac{\kappa}{2} \rfloor$ and $|S_2| = \lceil \frac{\kappa}{2} \rceil$. Since $\kappa \leq 2f$, we have $\lfloor \frac{\kappa}{2} \rfloor \leq f$ and $\lceil \frac{\kappa}{2} \rceil \leq f$, and so $\mathcal{S}_1$ and $\mathcal{S}_2$ are both legitimate candidate sets of malicious nodes (if one is interested in tolerating a maximum of f malicious nodes in the system). Thus, if $\kappa \leq 2f$, one cannot guarantee that all nodes can calculate any function of all initial node values when there are up to f malicious nodes in the system. ■

VI. Example

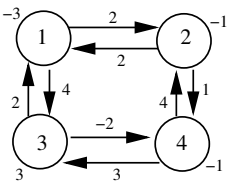

Fig. 1. Network with edge and self weights chosen from the set $\{-4, -3, -2, -1, 1, 2, 3, 4\}$.

Consider the network shown in Fig. 1. The objective in this network is for all nodes to calculate the function $g(x_1[0], x_2[0], x_3[0], x_4[0]) = \sum_{i=1}^{4} x_i^2[0]$. Since the network is strongly connected, Theorem 1 indicates that each node i can calculate any function of the initial values after running the linear iteration with almost any choice of weight matrix for at most $N - \deg_i = 2$ time-steps (when there are no malicious nodes in the network). For this example, we will choose each of the edge and self weights as an independent random variable uniformly distributed in the set[3] $\{-4, -3, -2, -1, 1, 2, 3, 4\}$. These weights are shown in Fig. 1, and produce the weight matrix

$$W = \begin{bmatrix} -3 & 2 & 2 & 0 \\ 2 & -1 & 0 & 4 \\ 4 & 0 & 3 & 3 \\ 0 & 1 & -2 & -1 \end{bmatrix} . \qquad (13)$$

One can verify that the observability matrix $\mathcal{O}_{i,N-\deg_i-1}$ is of full column rank (with rank 4) for each i, and thus each node can indeed obtain all initial values after $N - \deg_i = 2$ time-steps (via equation (3) with $Q = I_4$), and thereby calculate the function $\sum_{i=1}^{4} x_i^2[0]$.

However, suppose that we allow for the possibility that one or more of the nodes in the network are malicious. Since the network in Fig. 1 has connectivity $\kappa = 2$ (e.g., the set $\mathcal{S} = \{x_2, x_3\}$ forms a vertex cut), Theorem 2 indicates that only one malicious node is required in order to prevent some node from calculating the required function. For example, suppose that node 2 is malicious, and wants to prevent node 1 from obtaining any information about the value of node 4. Consider the weight matrix in (13). Since the nodes x_2 and x_3 form a vertex cut (separating the vertices x_1 and x_4), we see that the weight matrix is already in the form (8). Specifically, we have $W_{24} = 4$, $W_{34} = 3$ and $W_{44} = -1$. Suppose the initial values of the nodes are given by $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 2 & 1 \end{bmatrix}'$, and at each time-step, node 2 updates its value as

$$x_2[k+1] = 2x_1[k] - x_2[k] + 4x_4[k] + u_2[k] \ ,$$

where $u_2[k]$ is given by (11) with $\mathbf{a} = 1$ and $\mathbf{b} = -2$ (i.e., node 2 will attempt to prevent node 1 from determining whether $x_4[0] = 1$ or $x_4[0] = -2$). With this set of updates, the values seen by node 1 over the first 2 time-steps of the linear iteration are given by $\mathbf{y}_1[0] = \begin{bmatrix} 3 & -1 & 2 \end{bmatrix}'$ and $\mathbf{y}_1[1] = \begin{bmatrix} -7 & -1 & 21 \end{bmatrix}'$. However, one can verify that these are exactly the values seen by node 1 if the initial values were $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 2 & -2 \end{bmatrix}'$, and node 3 updates its values at each time-step as

$$x_3[k+1] = 4x_1[k] + 3x_3[k] + 3x_4[k] + u_3[k] \ ,$$

where $u_3[k] = W_{34}W_{44}^k(\mathbf{a} - \mathbf{b})$ with $\mathbf{a} = 1$ and $\mathbf{b} = -2$ (as specified in the proof of Lemma 1). Thus, node 1 cannot determine whether node 2 or node 3 is malicious, and thus cannot determine whether $x_4[0] = 1$ or $x_4[0] = -2$. As long as node 2 updates its values at each time-step with the errors given by (11), node 1 can never distinguish between malicious behavior by node 2 from malicious behavior by node 3, regardless of the number of time-steps for which the linear iteration is run. Node 2 has therefore succeeded in preventing node 1 from calculating its desired function.

[3] In general, the result in Theorem 1 will hold with high probability if one chooses the weights for the linear iteration from a continuous distribution over the real numbers (such as a Gaussian distribution). For this pedagogical example, however, it suffices to consider a distribution on a small set of integers.

VII. Summary

In this paper, we considered the problem of distributed function calculation in networks with malicious or malfunctioning nodes. Specifically, we studied a linear iterative strategy, and showed that while such a strategy allows nodes to calculate any function when there are no malicious nodes in the network, it is possible for a set of malicious nodes to update their values in such a way as to prevent some nodes from calculating any function of all node values. In particular, if the connectivity of the network is $2f$ or less, we showed that f malicious nodes can conspire to disrupt the network in this fashion. In the companion paper [11], we show that if the connectivity of the graph is greater than $2f$, the linear iterative strategy makes it impossible for f malicious nodes to prevent any node from calculating an arbitrary function of the initial values.

References

[1] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[2] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 20–27.

[3] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the American Control Conference*, 2005, pp. 1859–1864.

[4] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[5] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks*. Springer-Verlag, 2005.

[6] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[7] S. Sundaram and C. N. Hadjicostis, "Distributed consensus and linear functional calculation in networks: An observability perspective," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 99–108.

[8] ——, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proceedings of the American Control Conference*, 2007, pp. 711–716.

[9] ——, "Distributed function calculation and consensus using linear iterative strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, May 2008, to appear.

[10] C. N. Hadjicostis, *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*. Kluwer Academic Publishers, 2002.

[11] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterations in the presence of malicious agents – part II: Overcoming malicious behavior," in *Proceedings of the American Control Conference*, 2008.

[12] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 5594–5599.

[13] D. B. West, *Introduction to Graph Theory*. Prentice-Hall Inc., Upper Saddle River, New Jersey, 2001.

[14] C.-T. Chen, *Linear System Theory and Design*. Holt, Rinehart and Winston, 1984.